SPYWOLF

Security Audit Report

(TESTNET)

Completed on

April 27, 2023

OVERVIEW

This audit has been prepared for **nSights** to review the main aspects of the project to help investors make make an informative decision during their research process.

You will find a a summarized review of the following key points:

- ✔ Contract's source code
- ✔ Owners' wallets
- ✔ Tokenomics
- ✔ Team transparency and goals
- ✔ Website's age, code, security and UX
- ✔ Whitepaper and roadmap
- ✔ Social media & online presence

The results of this audit are purely based on the team's evaluation and does not guarantee nor reflect the projects outcome and goal

- SPYWOLF Team -

TABLE OF CONTENTS

nsights

PROJECT DESCRIPTION

nSights is the most advanced trading platform for new emerging tokens. Increase your chances of profitability within the complex world of Decentralized Finance (DeFi).

nSights demystifies DeFi and enhances the entire crypto trading experience. The nSights goal is to provide the previously inaccessible and necessary tools to help traders achieve their financial goals.

Release Date: Launched in October 2021

Category: Wallet

MIDWALLET CONTRACT INFO

Token Name
NSIMidWallet_V2

Symbol
N/A

Contract Address
0x50Bf76b05CB2f6b7E229fb23B59B2d6b13bc0a9C

Network
Binance Smart Chain
TESTNET

Language
Solidity

Deployment Date
Apr 18, 2023

Verified?
Yes

Total Supply
N/A

Status
Launched

TAXES

Buy Tax
n/a

Sell Tax
n/a

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

TOKEN TRANSFERS STATS

Transfer Count	TESTNET
Uniq Senders	TESTNET
Uniq Receivers	TESTNET
Total Amount	TESTNET
Median Transfer Amount	TESTNET
Average Transfer Amount	TESTNET
First transfer date	TESTNET
Last transfer date	TESTNET
Days token transferred	TESTNET

SMART CONTRACT STATS

Calls Count	TESTNET
External calls	TESTNET
Internal calls	TESTNET
Transactions count	TESTNET
Uniq Callers	TESTNET
Days contract called	TESTNET
Last transaction time	TESTNET
Created	TESTNET
Create TX	TESTNET
Creator	TESTNET

VULNERABILITY CHECK

Design Logic	Passed
Compiler warnings.	Passed
Private user data leaks	Passed
Timestamp dependence	Passed
Integer overflow and underflow	Passed
Race conditions and reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious Event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zeppelin module	Passed
Fallback function security	Passed

THREAT LEVELS

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and access control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Low Risk

Issues on this level are minor details and warning that can remain unfixed.

Informational

Information level is to offer suggestions for improvement of efficacy or security for features with a risk free factor.

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

NSIMidWallet_V2's owner can withdraw any tokens from the contract.

```
function clearStuckBnbBalance(uint256 amountPercentage) external onlyOwner {
    require(amountPercentage < 101, "Max 100%");
    uint256 amountBNB = address(this).balance;
    uint256 amountToClear = ( amountBNB * amountPercentage ) / 100;
    payable(msg.sender).transfer(amountToClear);
    emit BalanceClear(amountToClear);
}

function clearStuckToken(address tokenAddress, uint256 tokens) external onlyOwner returns (bool success) {
    require(tokenAddress != address(this), "MidWallet address not Allowed.");
    require(tokenAddress != address(0) , "Enter Non Zero Wallet Address.");
    require(Withdrawpaused == false, "Withdraw Paused.");
    require(disabled[tokenAddress] == false,
    "This token is disabled by the owner. Enable it again before withdrawing from the mid wallet.");

    if(tokens == 0){
        tokens = IERC20(tokenAddress).balanceOf(address(this));
    }
    emit clearToken(tokenAddress, tokens);
    return IERC20(tokenAddress).transfer(msg.sender, tokens);
}

function ownerWithdraw(uint256 _amount, address _tokenAddr) public onlyOwner {
    require(paused == false, "You can not withdraw while Swaping");
    require(Withdrawpaused == false, "Withdraw Paused.");
    require(disabled[_tokenAddr] == false, "This token is disabled by the owner. Enable it again before withdrawing from the mid wallet.");

    if(_tokenAddr == address(0)){
        payable(msg.sender).transfer(_amount);
    }else{
        IERC20(_tokenAddr).transfer(msg.sender, _amount);
    }
    emit Withdraw(owner, _amount, _tokenAddr);
}
```

FOUND THREATS

Informational

NSIMidWallet_V2's owner can disable tokens spending from the contract.

```
function disable(address[] memory tokenContractAddresses) public onlyOwner{
    if (tokenContractAddresses.length == 0) {
        return;
    }
    for (uint i = 0; i < tokenContractAddresses.length; i++) {
        enabled[tokenContractAddresses[i]] = false;
        disabled[tokenContractAddresses[i]] = true;
    }
}
```

NSIMidWallet_V2's owner can deposit any tokens in the contract.

```
function depositTokens(address _token , uint _amount) external checkAllowance(_token, _amount) onlyOwner {
    require(paused == false, "You can't depoist while Swaping");
    require(disabled[_token] == false,
    "This token is disabled by the owner. Enable it again before depositing in mid wallet.");
    IERC20(_token).transferFrom(msg.sender, address(this), _amount);
    emit Deposit(owner, _amount, _token);
}
```

FOUND THREATS

Informational

NSIMidWallet_V2's owner can approve 3rd party to spend contract's tokens.

```
function approveNewMidWallet(address newContractAddress, address[] memory tokens) external onlyOwner {
    // approve new smart contract to spend each enabled token
    for (uint i = 0; i < tokens.length; i++) {
        address tokenAddress = tokens[i];
        if (enabled[tokenAddress]) {
            IERC20 token = IERC20(tokenAddress);
            token.approve(newContractAddress, token.balanceOf(address(this)));
        }
    }
}
```

NSIMidWallet_V2's owner can transfer tokens from 3rd party in the contract, if necessary allowance is provided.

```
function transferTokensFromOldMidWallet(address oldContractAddress, address[] memory tokens) external onlyOwner{
    // check that user has approved new smart contract to spend each token
    for (uint i = 0; i < tokens.length; i++) {
        IERC20 token = IERC20(tokens[i]);
        uint256 allowance = token.allowance(oldContractAddress, address(this));
        require(allowance > 0, "New smart contract has not been approved to spend tokens");
    }

    // transfer tokens from old smart contract to new smart contract
    for (uint i = 0; i < tokens.length; i++) {
        IERC20 token = IERC20(tokens[i]);
        uint256 balance = token.balanceOf(oldContractAddress);
        if (balance > 0) {
        token.transferFrom(oldContractAddress, address(this), balance);
        }
    }
}
```

FOUND THREATS

Informational

NSIMidWallet_V2's owner and NSIMidWallet_V2's administrator can enable tokens for spending.

```
function enable(address[] memory tokenContractAddresses) public onlyOwnerOrAdmin {
    if (tokenContractAddresses.length == 0) {
        return;
    }
    for (uint i = 0; i < tokenContractAddresses.length; i++) {
        IERC20 token = IERC20(tokenContractAddresses[i]);
        token.approve(address(uniswapV2Router), type(uint256).max);
        enabled[tokenContractAddresses[i]] = true;
    }
}
```

NSIMidWallet_V2's administrator can pause tokens withdraw from the contract.

```
function WithdrawPaused(bool value) public onlyAdmin{
    Withdrawpaused = value;
}
```

FOUND THREATS

Informational

NSIMidWallet_V2's administrator can trade (swap) any tokens from the contract. Receiver of the swapped tokens is the wallet contract.

```
function Trade_Token(address _tokenIn, address _tokenOut, uint256 _amountIn,
address[] memory path,  uint256 _amountOutMin) public onlyAdmin{
    paused ==  true;
    address _to = address(this);
    address  WETH = uniswapV2Router.WETH();
    if(_tokenIn == WETH){
        require(enabled[_tokenOut] == true,"The token you are trying to swap is not enbaled in MidWallet.");
    }else if(_tokenOut == WETH){
        require(enabled[_tokenIn] == true,"The token you are trying to swap is not enbaled in MidWallet.");
    }else{
        require(enabled[_tokenIn] == true && enabled[_tokenOut],"The token you are trying to swap is not enbaled in MidWallet.");
    }
    uint deadline = block.timestamp + 3600;
    uint[] memory amounts = IUniswapV2Router02(uniswapV2Router).swapExactTokensForTokens(_amountIn, _amountOutMin, path, _to, deadline);
    uint256 amountOut = amounts[amounts.length - 1];
    paused == false;

    emit Resultant_Token(owner, amountOut, _tokenOut);

}

function Trade_Token_Supporting_Fee(address _tokenIn, address _tokenOut,
uint256 _amountIn, address[] memory path, uint256 _amountOutMin) public onlyAdmin{
    paused == true;
    address _to = address(this);
    address  WETH = uniswapV2Router.WETH();
    if(_tokenIn == WETH){
        require(enabled[_tokenOut] == true,"The token you are trying to swap is not enbaled in MidWallet.");
    }else if(_tokenOut == WETH){
        require(enabled[_tokenIn] == true,"The token you are trying to swap is not enbaled in MidWallet.");
    }else{
        require(enabled[_tokenIn] == true && enabled[_tokenOut],"The token you are trying to swap is not enbaled in MidWallet.");
    }
    uint deadline = block.timestamp + 3600;
    IUniswapV2Router02(uniswapV2Router).swapExactTokensForTokensSupportingFeeOnTransferTokens(_amountIn, _amountOutMin, path, _to, deadline);
    paused == false;
}
```

FOUND THREATS

Informational

IMPORTANT
NSIMidWalletFactory_V2's owner has administrator rights of every NSIMidWallet_V2 contract derived from the factory.

MIDWALLETFACTORY CONTRACT INFO

Token Name NSIMidWalletFactory_V2	Symbol N/A
Contract Address 0x13500FCE07960611b29DcD54267a1150Fc92892A	
Network Binance Smart Chain TESTNET	Language Solidity
Deployment Date Apr 18, 2023	Verified? Yes
Total Supply N/A	Status Launched

TAXES

Buy Tax	Sell Tax
n/a	n/a

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

TOKEN TRANSFERS STATS

Transfer Count	TESTNET
Uniq Senders	TESTNET
Uniq Receivers	TESTNET
Total Amount	TESTNET
Median Transfer Amount	TESTNET
Average Transfer Amount	TESTNET
First transfer date	TESTNET
Last transfer date	TESTNET
Days token transferred	TESTNET

SMART CONTRACT STATS

Calls Count	TESTNET
External calls	TESTNET
Internal calls	TESTNET
Transactions count	TESTNET
Uniq Callers	TESTNET
Days contract called	TESTNET
Last transaction time	TESTNET
Created	TESTNET
Create TX	TESTNET
Creator	TESTNET

VULNERABILITY CHECK

Design Logic	Passed
Compiler warnings.	Passed
Private user data leaks	Passed
Timestamp dependence	Passed
Integer overflow and underflow	Passed
Race conditions and reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious Event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zeppelin module	Passed
Fallback function security	Passed

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Contract’s administrators can create new wallets for users.

```
function createFor(address owner, address[] memory enableTokenAddresses)
public onlyAdmin returns (address created) {
    if (owners[owner] != address(0)) {
        return owners[owner];
    }

    NSIMidWallet_V2 newContract = new NSIMidWallet_V2(owner);
    //Create the New Mid Wallet against owner address
    if (enableTokenAddresses.length > 0) {
        newContract.enable(enableTokenAddresses);
        // Enable the tokens in the new contract
    }
    owners[owner] = address(newContract);
    return owners[owner];
}
```

ACCESSPASS CONTRACT INFO

Token Name
AccessPass

Symbol
N/A

Contract Address
0x1d6989141862388a1edccb801e07b8d8687d8d36

Network
Binance Smart Chain
TESTNET

Language
Solidity

Deployment Date
Apr 18, 2022

Verified?
Yes

Total Supply
N/A

Status
Launched

TAXES

Buy Tax
n/a

Sell Tax
n/a

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

TOKEN TRANSFERS STATS

Transfer Count	TESTNET
Uniq Senders	TESTNET
Uniq Receivers	TESTNET
Total Amount	TESTNET
Median Transfer Amount	TESTNET
Average Transfer Amount	TESTNET
First transfer date	TESTNET
Last transfer date	TESTNET
Days token transferred	TESTNET

SMART CONTRACT STATS

Calls Count	TESTNET
External calls	TESTNET
Internal calls	TESTNET
Transactions count	TESTNET
Uniq Callers	TESTNET
Days contract called	TESTNET
Last transaction time	TESTNET
Created	TESTNET
Create TX	TESTNET
Creator	TESTNET

VULNERABILITY CHECK

Design Logic	Passed
Compiler warnings.	Passed
Private user data leaks	Passed
Timestamp dependence	Passed
Integer overflow and underflow	Passed
Race conditions and reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious Event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zeppelin module	Passed
Fallback function security	Passed

FOUND THREATS

⚠ High Risk

withdrawableAmount is deducted twice from user's TokensDeposited. Overflow may occur when autoWithdraw is initiated.

```
function autoWithdraw(address[] memory addresses) external {

require(msg.sender == owner, "Only Owner");
for (uint i = 0; i < addresses.length; i++) {
    address user = addresses[i];
    if (userProfile[user].autoWithdraw && canWithdraw(user) && userProfile[user].status ) {
        require(userProfile[user].TokensDeposited > 0,"Insufficient funds to withdraw.");
        require(canWithdraw(user), "Not eligible to withdraw.");
        require(userProfile[user].status,"User is Inactive.");
        uint256 currentPricePerToken = currentTokenPrice();
        uint256 UserDepositTokens = userProfile[user].TokensDeposited;
        uint256 UserDepositLimitBUSD = userProfile[user].totalPriceBUSD;
        uint256 currentTotalPrice = getAmountOutMin(BUSD, currentToken, UserDepositLimitBUSD);
        uint256 withdrawableAmount =  UserDepositTokens - currentTotalPrice;
        IERC20(currentToken).transfer(user, withdrawableAmount);
        userProfile[user].TokensDeposited -= withdrawableAmount;
        userProfile[msg.sender].latestTrxPrice = currentPricePerToken;
        userProfile[user].TokensDeposited -= withdrawableAmount;
        userProfile[user].trxTime = block.timestamp;
        userDetailsHistory.push(userDetails(user,userProfile[user].TokensDeposited,
        userProfile[user].accessPassedTokenPrice,currentPricePerToken,userProfile[user].totalPriceBUSD,
        block.timestamp,userProfile[msg.sender].worthIncrease ,true, true, userProfile[msg.sender].autoWithdraw));
        emit Withdraw(msg.sender, withdrawableAmount, currentToken);
    }
}

}
```

FOUND THREATS

Informational

Owner can set min busd value deposit limit and token increase percent criteria.

```
function setDepositLimitInBusd(uint256 busdAmount) external {
    require(msg.sender == owner , "Only Owner");
    currentDepositBusdLimit = busdAmount * 10 **18;
}

function updateTokenWorthIncrease(uint256 percentage) external {
    require(msg.sender == owner, "Only Owner");
    tokenWorthIncrease = percentage;
}
```

Owner can change users' auto withdraw status and user's status. In order for users to use the userWithdrawProfit() function, their status must be active.

```
function updateAutoWithdrawStatus(address[] memory users, bool autoWithdrawStatus) external {
    require(msg.sender == owner, "Only Owner");
    require(users.length < 501, "Can not pass more than 500 addresses.");
    for (uint i= 0; i< users.length; i++) {
        userProfile[users[i]].autoWithdraw = autoWithdrawStatus;
    }
}

function updateUserStatus(address[] memory users, bool active) external {
    require(msg.sender == owner, "Only Owner");
    require(users.length < 501, "Can not pass more than 500 addresses.");
    for (uint i = 0; i< users.length; i++ ) {
        userProfile[users[i]].status = active;
    }
}
```

FOUND THREATS

Informational

Owner can withdraw any tokens from the contract.

```
function updateCurrentToken(address newToken) external {
    require(msg.sender == owner, "Only Owner");
    currentToken = newToken;
}

function recoverStuckBnb(uint256 amountPercentage) external  {
    require(msg.sender == owner, "Only Owner");
    require(amountPercentage < 101, "Max 100%");
    uint256 amountBNB = address(this).balance;
    uint256 amountToClear = ( amountBNB * amountPercentage ) / 100;
    payable(msg.sender).transfer(amountToClear);
    emit BalanceClear(amountToClear);
}

function recoverStuckToken(address tokenAddress, uint256 tokens) external  returns (bool success) {
    require(msg.sender == owner, "Only Owner");
    require(tokenAddress != currentToken , "You can not Withdraw this Token.");
    require(tokenAddress != address(0) , "Enter Non Zero Wallet Address.");
    if(tokens == 0){
        tokens = IERC20(tokenAddress).balanceOf(address(this));
    }
    emit clearToken(tokenAddress, tokens);
    return IERC20(tokenAddress).transfer(msg.sender, tokens);
}
```

FOUND THREATS

Informational

Owner can nullify deposit and return the deposited tokens to the depositor.

```
function transferUsersBalanceToOwner(address[] memory inactiveUsers) external {
    require(msg.sender == owner, "Only Owner");
    require(inactiveUsers.length > 0,"Pass at least one inactive user address");
    for (uint i = 0; i < inactiveUsers.length; i++) {
        address InActiveUser = inactiveUsers[i];
        require(userProfile[InActiveUser].TokensDeposited > 0, "The user does not have any funds to transfer.");
        require(!userProfile[InActiveUser].status, "The status of user is Still active");
        uint256 userBalance = userProfile[InActiveUser].TokensDeposited;
        IERC20(currentToken).transfer(InActiveUser, userBalance);
        userProfile[InActiveUser].userAddress = InActiveUser;
        userProfile[InActiveUser].TokensDeposited -= userBalance;
        userProfile[InActiveUser].accessPassedTokenPrice = 0;
        userProfile[InActiveUser].latestTrxPrice = 0;
        userProfile[InActiveUser].totalPriceBUSD = 0;
        userProfile[InActiveUser].trxTime = block.timestamp;
        userProfile[InActiveUser].hasDeposited = false;
        userProfile[InActiveUser].status = false;
        userProfile[InActiveUser].autoWithdraw = false;
        userDetailsHistory.push(userDetails(InActiveUser,0,0,0,0,block.timestamp, 0,false, false, false));
        inactiveUserTransferHistory.push(userDetails(InActiveUser, userBalance,0,0,0,block.timestamp, 0, false, false, false));
        emit Transfer(InActiveUser, owner, userBalance);
    }
}
```

ABOUT US

We are a growing crypto security agency offering audits, KYCs and consulting services for some of the top names in the crypto industry.

- **OVER 150 SUCCESSFUL CLIENTS**
- **MORE THAN 500 SCAMS EXPOSED**
- **MILLIONS SAVED IN POTENTIAL FRAUD**
- **PARTNERSHIPS WITH TOP LAUNCHPADS, INFLUENCERS AND CRYPTO PROJECTS**
- **CONSTANTLY BUILDING TOOLS TO HELP INVESTORS DO BETTER RESEARCH**

SPYWOLF

CRYPTO SECURITY

Audits | KYCs | dApps
Contract Development

To hire us, reach out to
contact@spywolf.co or
t.me/joe_SpyWolf

FIND US ONLINE

SPYWOLF.CO

SPYWOLF.NETWORK

@SPYWOLFNETWORK

@SPYWOLFOFFICIAL

@SPYWOLFNETWORK

@SPYWOLFNETWORK

Disclaimer

This report shows findings based on our limited project analysis, following good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall social media and website presence and team transparency details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report.
While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER:
By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice.
No one shall have any right to rely on the report or its contents, and SpyWolf and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (SpyWolf) owe no duty of care towards you or any other person, nor does SpyWolf make any warranty or representation to any person on the accuracy or completeness of the report.
The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SpyWolf hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SpyWolf hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SpyWolf, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts, website, social media and team.
No applications were reviewed for security. No product code has been reviewed.